NEOTERIC

15816 SAN BENITO WAY
LOS GATOS, CA 95030
(408) 356-6477

COMPUTER SYSTEM DESIGN

March 17, 1980

Mr. George Simcock
Atari, Inc.
1272 Borregas Avenue
Sunnyvale, CA 94086

Dear Mr. Simcock:

Enclosed are some comments I have regarding the Atari Pascal being developed by MT MicroSYSTEMS, as defined in the Atari Pascal Software Functional Specification. Where information was missing, I referred to: my notes from prior meetings, the Meta Tech proposal of November 9, 1979 and the Pascal/MT User's Guide to Release 2.0, in order to deduce Meta Tech's intent.

I did not attempt to evaluate their P-Code instruction set, being presently concerned primarily with the language definition and the system operation.

I can well understand their not having all of the required documentation at this time, but I do think Atari should have at least an informal document or letter addressing the questions raised. The documentation received thus far has been well organized and error free, and I have every reason to believe that the resultant software will be of similar quality.

Yours very truly,

Harry B. Stewart

Notes on Atari Pascal Sofware Functional Specification

GENERAL COMMENTS

After having done a lot of "systems level" programming in Pascal, I am glad to see the inclusion of the BYTE and WORD data types, the hexadecimal radix specifier '$', and the ability to perform arithmetic on a pointer without the use of a case variant record. The bit twiddling procedures and functions should prove quite useful, too.

The separately compiled MODULE capability will really help out our users, because of the slow disk I/O they will be faced with.

I am a little concerned about static variable allocation, and I don't yet understand all of its implications; for example, it appears that the minimum RAM required to run will be the sum of the RAM required by each individual procedure and function (not the case in standard Pascal). More obviously, procedures and functions may not recurse except in a most trivial manner (as in a BASIC GOSUB).

Apparently native code elements (such as the run-time support package) can be linked to a native code Pascal program, but it is not clear at what level this occurs and whether this capability can be extended to include user defined native code procedures and functions. We need more information regarding this capability.

PASCAL LANGUAGE EXCEPTIONS

The specification lists the Pascal language extensions to be supported, but does not list any language exceptions; does this mean that there are no language exceptions? At earlier meetings, the following exceptions were identified:

1. Enumeration was not to have been supported.

```
TYPE colors = (blue,red,green,yellow,brown,orange,purple);
```

2. Sets were not to have been supported.

```
TYPE palette = SET OF colors;
TYPE alpha = SET OF letters;
```

Are there other exceptions? If so, we need to know them.

PASCAL IMPLEMENTATION EXCEPTIONS

There were some implementation exceptions discussed at earlier meetings which were not mentioned in the specification; what is

the status of the following?

1. A PACKED ARRAY was to have been synonymous with an unpacked ARRAY. The PACK and UNPACK standard procedures would have no function. Supposedly CHAR arrays were to be always packed, BOOLEAN arrays always unpacked and sub-range types were not mentioned.

2. All variables and parameters were to have been allocated statically (compile time rather than run time).

Are there other exceptions? If so, we need to know them.

PASCAL LANGUAGE EXTENSIONS

Several extensions were agreed upon at earlier meetings and have not yet been documented:

1. A pointer variable may be used anywhere an integer variable may be used.

```
VAR mempoint:^INTEGER;
    i:INTEGER;

mempoint := $4000;
i := mempoint^;
mempoint := mempoint + 2;
```

2. A memory location assignment was to have been optional in variable declarations; the syntax was not specified.

3. Although used in many examples, nowhere does it state that the underscore character is allowed in (all?) identifiers.

```
PROCEDURE get_record;
```

4. Hexadecimal radix was allowed in integer specifications in the Pascal/MT documentation we received; is it supported by Atari Pascal?

```
CONST all_ones = $FFFF;
invert_char := old_char + $80;
```

Also, are there any accommodations to hexadecimal in the I/O subsystem?

MISSING PASCAL DOCUMENTATION:

The functional specification, while it covered many areas in great detail, covered other areas either not at all or too briefly. The topics below deserve more attention.

1. All user/program dialogue for the Compiler,Linker and Emulator programs needs to be specified.

Process initiation by user.
Program prompting to the user.
Program error messages.

2. The Compiler directives need to be listed and described.

3. The system response to run-time errors for both the emulator and the native-code run-time library needs to be defined.

4. The user file data format must be described (BASIC compatible?).

5. We need a definition of what system resources a native code program will require to run.

Atari Floating Point Package ROM?
CIO?
O.S. Data Base [0000-047F]?
etc.?

6. It was stated at one of the meetings that type checking would not be rigidly adhered to; therefore we need a definition of when type checking is provided and when it is not, plus transformation rules for assignments and comparisons when types don't match.

7. A discussion of range checking is in order: where it applies, and whether it may be disabled as a compile time or run-time option.

8. What restrictions does the limited 6502 hardware stack size (256 bytes) place on Pascal programs in the emulator and the native code environments?

9. NEW, DISPOSE and MEMAVAIL need a final definition.

10. Are RESET, EOF and EOLN implemented as per standard Pascal? Are any exceptions made for the interactive console?

11. While not essential now, it would be helpful to have a brief description of the emulator's machine resource utilization, for design review purposes.

ADDITIONAL PROPOSED CHANGES

In addtition to those extensions already committed to by MT MicroSYSTEMS, I would like to have the following considered:

1. EXIT <name>, ala UCSD Pascal.

2. Have any of the proposed Pascal standards allowed for iterations to be performed without the use of a control variable, such as shown below?

FOR 1 TO 5 DO <block>;

or

REPEAT <block> 23 TIMES;

If so, I suggest we consider adopting this capability.